Application Note :
Telink USB Module User Guide

AN-18080200-E1

Ver 1.0.0

2018/8/7

Brief:

This document is the user guide for Telink USB module.

TELINK SEMICONDUCTOR

Published by
Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,
Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

Telink Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein or in any other disclosure relating to any product.

Telink Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others

The products shown herein are not designed for use in medical, life-saving, or life-sustaining applications. Customers using or selling Telink Semiconductor products not expressly indicated for use in such applications do so entirely at their own risk and agree to fully indemnify Telink Semiconductor for any damages arising or resulting from such use or sale.

Information:

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Revision History

Version	Major Changes	Date	Author
1.0.0	Initial release	2018/8	ZXD, SGJ, Cynthia

Table of contents

Table of figures

Table of tables

1 Overview

The figure below shows block diagram for the USB module.

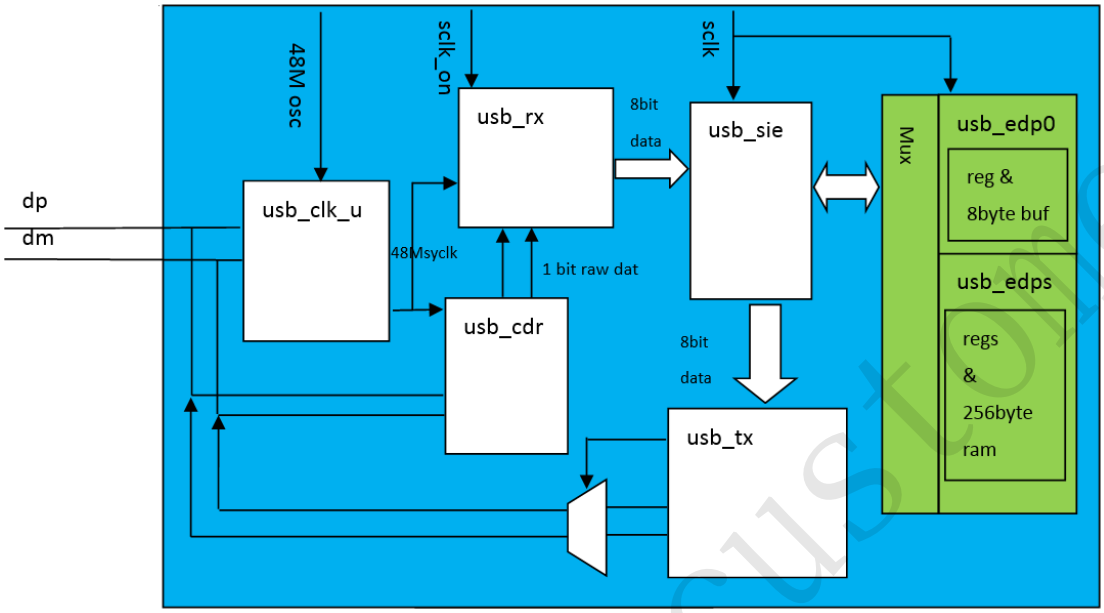

Figure 1 Block diagram of USB module

*Note: usb_edps corresponds to "edp1~edp8".

The USB module uses clock signals of three types, including 48M osc (internal 48MHz osc clock), sclk (system clock) and 48Msyclk (48MHz synchronized clock).

The usb_cdr module uses the 48Msyclk to get raw USB data via analysis, and the output data is then sent to the usb_rx module.

The usb_rx module supports two clock domain, i.e. 48Msyclk and sclk. First the usb_rx module implements NRZI decoding for the raw USB data and removes fill code 0 to assemble the data into data bytes. Then the data will be sent to the usb_sie module.

The usb_sie module implements preliminary analysis to basic data including PID, EPD_ADR and TOKEN and automatically generates ACK/NAK packet (sent via the usb_tx module). The useful data part will be provided to the usb_edp0/usb_edps.

The usb_edp0 contains a description table of usb printer device, which can be automatically enumerated as usb printer device for device debugging. Manual mode

is also selectable to enumerate it as user-defined device. An internal 8-byte space is available to buffer data sent from or received by the edp0.

For the convenience of further program processing, the usb_edp1~usb_edp8 can store data into the internal RAM or read data from the internal RAM. Note that corresponding to PC, the endpoints including edp1~edp4 and edp7~edp8 can only act as input, and the endpoints including edp5~edp6 can only act as output.

2 Register Table

Table 1 Register table for USB module

Address	Reg	Description	Reset Value
0x0100	EDP0PTR	Endpoint 0 buffer point	RW(0)
0x0101	EDP0DAT	Endpoint 0 buffer data access address	RW(XX)
0x0102	EDP0CT	bit[0]: Ack data bit[1]: Stall data bit[2]: Ack status bit[3] Stall status	
0x0103	EDP0ST	bit[3:0]: number of data transferred bit[4]: setup interrupt flag bit[5]: data interrupt flag bit[6]: status interrupt flag bit[7]: set interface interrupt flag	RO
0x0104	EDP0MODE	[0]: enable auto decoding set_address command [1]: enable auto decoding set_config command [2]: enable auto decoding set_interface command [3]: enable auto decoding get_status command [4]: enable auto decoding sync_frame command [5]: enable auto decoding get_descriptor command [6]: enable auto decoding set_feature command [7]: enable auto decoding standard command	RW(0xff)
0x0105	USBCT	[0]: use auto calibrate clock if 1, use system clock if 0 [1]: low speed mode if 1; full speed mode if 0 [2]: low jitter mode if 1; [3]: usb test mode	RW(0x01)
0x0106	__RESERVED		
0x0107	__RESERVED		
0x010a	MDEV	[0]: self power [1]: SUB suspend status	RO

Address	Reg	Description	Reset Value
		[2]: wakeup feature status	
0x010b	USBADR	bit[6:0]: usb adress bit[7]: config_now	RO
0x010c	SUSPENDCYC	bit[7:0]: suspend_cnt	RW(0x18)
0x010d	INTERFACE	bit[3:0]: set interface data now bit[7:4]: set interface buff idx	RO
0x0110	EDP8PTR		RW
0x0111	EDP1PTR		RW
0x0112	EDP2PTR		RW
0x0113	EDP3PTR		RW
0x0114	EDP4PTR		RW
0x0115	EDP5PTR		RW
0x0116	EDP6PTR		RW
0x0117	EDP7PTR		RW
0x0120	EDP8CT	[0]: ACK [1]: Stall [2]: Set Data0 [3]: Set Data1 [7]: Launch EOF for FIFO mode	
0x0121	EDP1 CT	[0]: ACK [1]: Stall [2]: Set Data0 [3]: Set Data1	
0x0122	EDP2 CT		
0x0123	EDP3 CT		
0x0124	EDP4 CT		
0x0125	EDP5 CT		
0x0126	EDP6 CT		
0x0127	EDP7 CT		
0x0128	EDP8 ADR	Endpoint 8 buffer address	RW(0x00)
0x0129	EDP1 ADR	Endpoint 1 buffer address	RW(0x08)
0x012a	EDP2 ADR	Endpoint 2 buffer address	RW(0x10)
0x012b	EDP3 ADR	Endpoint 3 buffer address	RW(0x40)
0x012c	EDP4 ADR	Endpoint 4 buffer address	RW(0xC0)
0x012d	EDP5 ADR	Endpoint 5 buffer address	RW(0x20)
0x012e	EDP6 ADR	Endpoint 6 buffer address	RW(0x30)
0x012f	EDP7 ADR	Endpoint 7 buffer address	RW(0x18)
0x130	USBRAM	[0]: CEN in power down mode [1]: CLK in power down mode [2]: Reserved [3]: WEN in power down mode	RW(0x00)

Address	Reg	Description	Reset Value
		[4]: CEN in function mode	
0x138	USBSO	Enable endpoint ISO mode bit[0]: Endpoint 8 iso mode bit[1]: Endpoint 1 iso mode … bit[7]: Endpoint 7 iso mode	RW(0xc0)
0x139	USBIRQ	bit[0]: Endpoint 8 irq bit[1]: Endpoint 1 irq … bit[7]: Endpoint 7 irq	RW(0x00)
0x13a	USBMASK	irq mask, same as USBIRQ	RW(0xff)
0x13b	USBMAX8	the max buff size for Endpoint 8: max_size = {USBMAX0[5:0],3'h0}	RW(0x10)
0x13c	USBMIN8	the min data in buff to send as a packet: the buff must have USBMIN8 data to ack to IN TOKEN	RW(0x40)
0x13d	USBFIFO	bit[0]: Endpoint 8 fifo mode: the pointer of Endpoint8 auto as a circuit buffer bit[1]: fifo full status bit[7:2]: dma mode for Endpoint8(NO USED FOR MATRIX)	RW(0x01)
0x13e	USBMAX	bit[7:0]: max data in for Endpoint buffer(execept 7) max data size = USBMAX*8	RW(0x08)
0x13f	USBTICK	bit[7:0]: just a tick that increase on posedge of the sclk_usb	RW(0)

3 USB Transaction

The figure below shows an USB transaction, e.g. get device description.

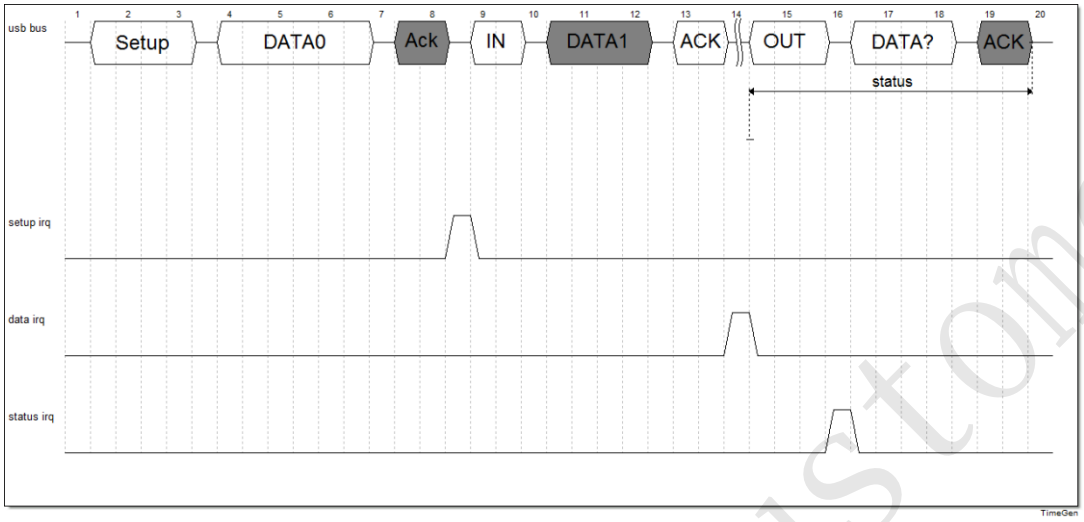

Figure 2 USB transaction

*Note: The part marked in black means packet sent by device. The interrupt above is Endpoint0 interrupt, and interrupt of other Endpoints is similar to data irq.

Data transmission and reception mechanism is shown as below:

When PC sends an IN/OUT token, if the ACK bit in the EDP_CT is set, data will be fetched from buffers corresponding to various endpoints and sent to PC (IN), or receive data from PC and store the data into buffers (OUT). Then an ACK signal will be returned (OUT); if the corresponding ACK bit is not set, NAK packet will be returned to inform PC to re-transmit the data later.